ANDROID DEVELOPMENT #7

@brmlab

NDK

- C / C++
- Interfaces to Java classes definition
- JNI to integrate with Java application
- OpenGL C++

NDK PURPOSE

- Not suitable for every app development
- Not necessarily faster than Java (relative)
- 2D / 3D game engines
- Pre-implemented algorithms
- 3RD party (media codecs, crypto, opencv)

ANDROID GAME ENGINES

- Cocos2D-X
- OGRE
- PowerVR
- Marmalade
- AndEngine
- LINDERDAUM

APPS USING NDK ONLY?

YES (with limits)

JNI

- Java Native Interface
- Not only “Android thing”
- Bi-Directional API
- Javac for SDK, GCC/G++ for NDK

JNI USAGE

- [Java] System.loadLibrary(String)
- [Java] keyword “native”
  - native {type} functionName({args})
  - eg. native String getUname();
- [C/C++] jclass, jstring, jobject, jint
- [C/C++] JNIEnv, JavaVM

NDK DEBUGGING

- Same debug process as with SDK
- Breakpoints, Expressions, Stacktraces
- Separated from SDK
  - separately raised exceptions and debugging
  - can’t trace in hybrid java/native complex

NDK PERMISSIONS

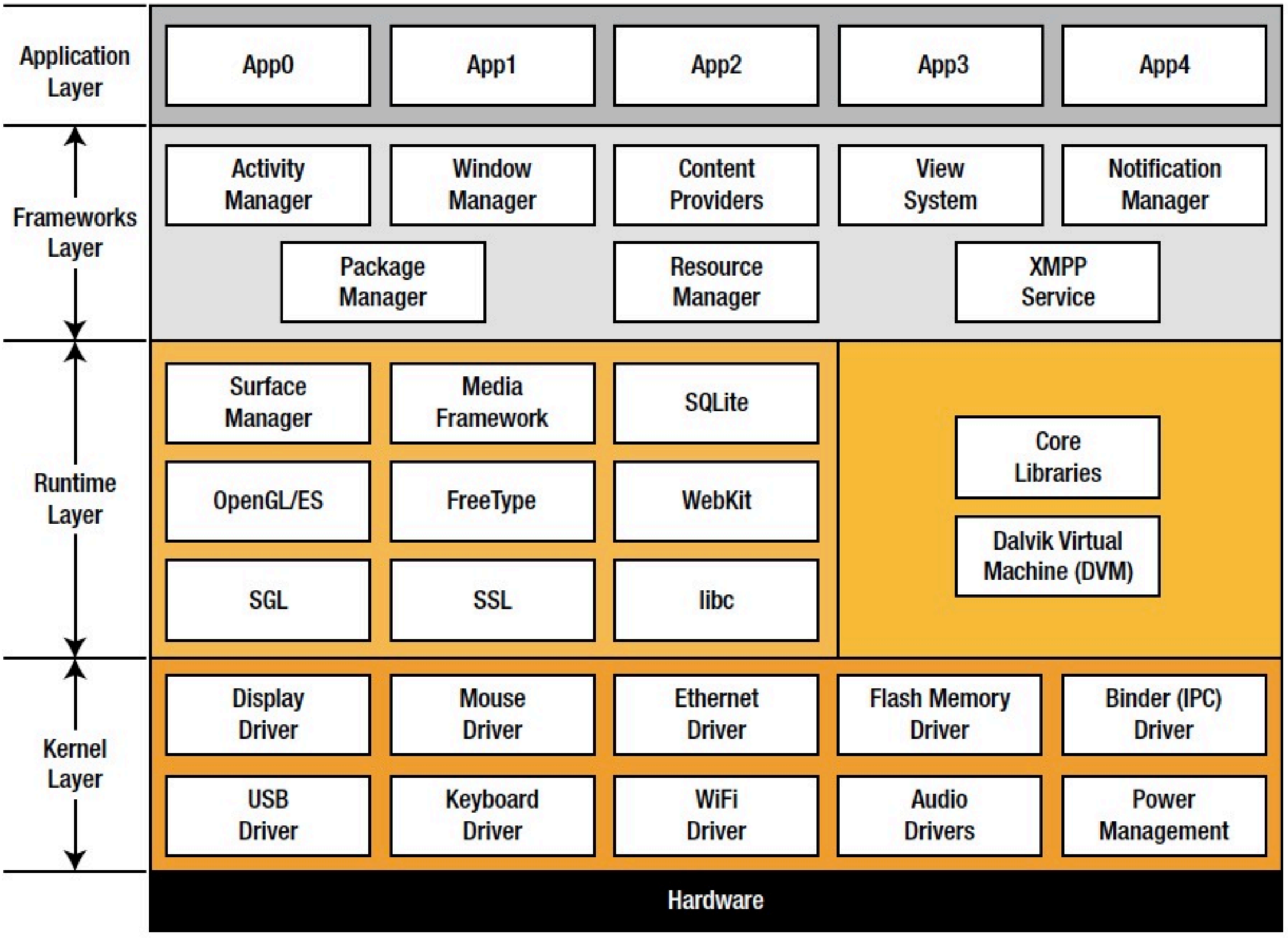
- No special NDK permissions
- Permissions inherited from AndroidManifest.xml

NDK SECURITY

- Kernel-level sandboxing
- Java VM Zygote - Single control process
- More vulnerable than SDK (app->system)
- Most ROM cracked through NDK/JNI

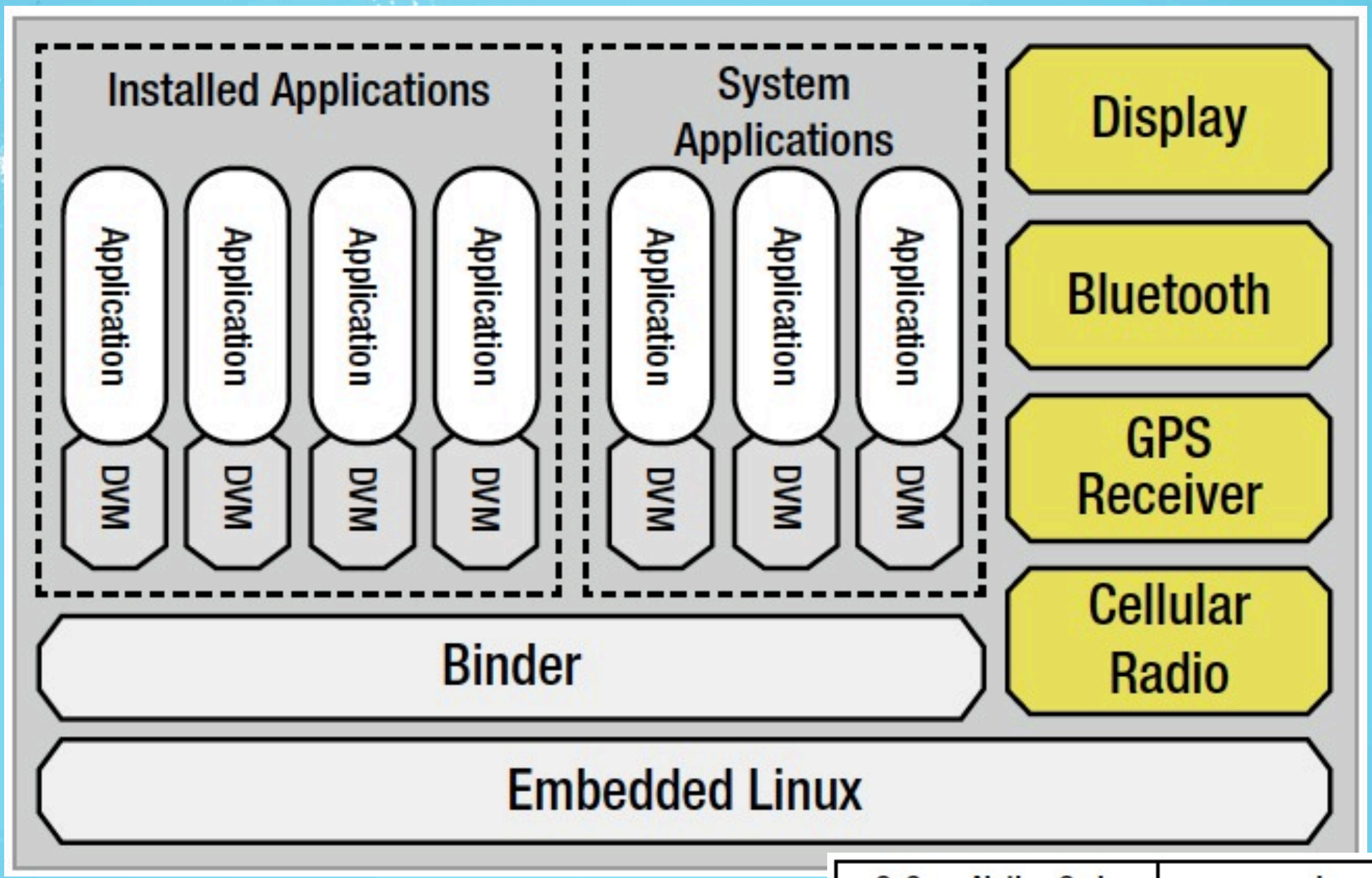
ANDROID SECURITY

- Application Sandboxing
- Process runtime separation (POSIX)
- Applications are given unique UID and GID
- Zygote controls resources/libraries usage
- APK is signed (SSL manner)
- Application permissions


Application Layer
App0
App1
App2
App3
App4
Frameworks Layer
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Resource Manager
XMPP Service
Runtime Layer
Surface Manager
Media Framework
SQLite
OpenGL/ES
FreeType
WebKit
SGL
SSL
libc
Core Libraries
Dalvik Virtual Machine (DVM)
Kernel Layer
Display Driver
Mouse Driver
Ethernet Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keyboard Driver
WiFi Driver
Audio Drivers
Power Management
Hardware

APPLICATION PERMISSIONS

- Enforced on Framework level
- Cannot be changed on runtime
- Both Hardware and Software permissions
- Application can create new permissions



C, C++, Native Code	Java
= Linux Kernel	= Android Frameworks
= Libraries	= Applications
= Android Runtime	

PERMISSIONS PROTECTION

- On application defined permissions protectionLevel
  - 0 - normal - no special confirmation
  - 1 - dangerous - may require confirm.
  - 2 - signature - required signature match
  - 3 - signatureOrSystem - special

DEVICE ADMINISTRATION

- Special type of permission
- Antivirus apps (ESET, Lookout)
- Lock, Wipe, Password (renewal, change, expire,...)
- Disable camera, Encrypt storage,
- Requires standalone confirmation
- since API 9

ANDROID VULNERABILITY

- SharedPreferences are plain XML
- SQLite databases are plain .sqlite files
- Java is easily decompilable (JD)
- Android DEX is too (Dex2JAR, dedexer)
- strings, layouts, drawables are plain xml/bitmap
- MITM predisposition

SECURING APPLICATIONS

- ProGuard (obfuscating code)
- In-App-Billing security (LVL)
- OAuth (OpenID, Google APIs)
- Including SSL certificates in app bundle (not relying on system provided ones)

SECURING SOLUTION

- Secure your API
  - API is not hidden from attackers
- Secure sensitive data in databases and storages (asymmetric crypto is your friend)
- You cannot just include key-pair
- You cannot hide passwords in code

JAVA HEAP

- Applications are given memory limit
- Differs by device, api level and hw params
  - typically between 16 and 32 MB
  - tablet PCs and high-ends have more
- Exceeding leads to OutOfMemoryError
- Beware of (bad) working with bitmaps

MEMORY LEAKS

- Garbage Collector is not a solution
- Context memory leaks
- Objects are assigned context but not cleaned up
- Beware of static class member
- Beware of attaching to application context

INDICATION

D/dalvikvm(1325): GC_CONCURRENT freed 1971K, 18% free 12382K/14983K, paused 3ms+7ms

Heap statistics

Reason for garbage collection:

```
GC_CONCURRENT
GC_FOR_MALLOC
GC_EXTERNAL_ALLOC
GC_HPROF_DUMP_HEAP
GC_EXPLICIT
```

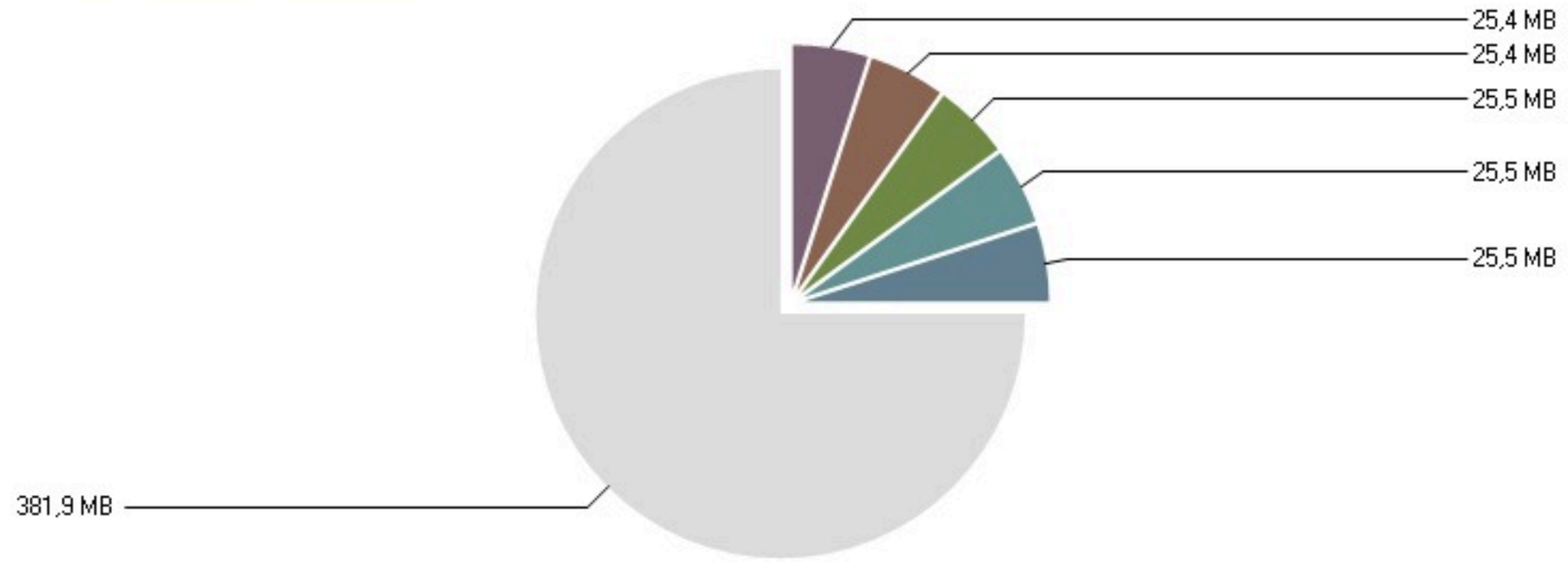
LEAK DESTRUCTION

- dump HPROF
  - Heap Allocation Profile
- Eclipse MAT
  - Memory Analyzer Tool

Details

Size: **509,3 MB** Classes: **7,1k** Objects: **19,1m** Class Loader: **138**

Biggest Objects by Retained Size



Total: 509,3 MB

Remainder

Actions

Histogram: Lists number of instances per class

Dominator Tree: List the **biggest objects** and what they keep alive.

Top Consumers: Print the most **expensive objects** grouped by class and by package.

Duplicate Classes: Detect classes loaded by multiple class loaders.

Reports

Leak Suspects: includes leak suspects and a system overview

Top Components: list reports for components bigger than 1 percent of the total heap.

Step By Step

Component Report: Analyze objects which belong to a **common root package** or **class loader**.